Title

Cisp: a Live-Coding language for Non-Standard Synthesis Algorithms

Abstract

My research is focused on developing a live-coding language for algorithmic composition called Cisp[1]. Cisp is a Lisp-lookalike that is transpiled into ChucK[2] code and was inspired by the work of Paul Berg (*PILE*[3]) and Luc Döbereiner (*CompScheme*[4]). Like these examples, the language has a simplified method of sound synthesis: sound is understood as a stream of amplitude values. For example, Cisp can generate amplitude streams by sequencing lists of values or generating their output using random number generators. Complexity is achieved by combining and processing these streams, using methods including: sequencing, indexing, iteration, interpolation, looping, reading and writing from memory etc..

One of Cisp's features is that the user can replace part of the program while it is running, with no interruption in sound. Parameters that define the behavior of streams can either be literal static values or controlled by other streams. The iteration cycle of rewriting programs in Cisp (write a program, listen, rewrite, listen, rewrite etc..), has opened up a different approach than I usually had writing sound algorithms, since Cisp programs are typically rewritten in very small steps. In this way, instead of making a plan and translating that idea into code (as I would do in C or SuperCollider), I often start with a very small program and keep rewriting parts of it, guided by the current aural state.

Proposal for presentation:

I propose I give a short presentation of 25 minutes (15 minutes presentation, 10 minutes discussion), that consists of:

An overview of how the Cisp transpiler developed, what motivated me to use ChucK as a basis and what other considerations were put in the design of the language.

After this, I would like to present a short live-coding session, using heavily commented code, demonstrating how a typical live coding session evolves.

[1] https://github.com/casperschipper/cisp , example programs can be found there as well
[2] https://chuck.cs.princeton.edu/
[3] https://www.jstor.org/stable/3679754?seq=1
[4] https://zenodo.org/record/849515#.XkmXjkMo8Wo

Finally, I would like to open the discussion for limitations of this system and possible areas of extension. Questions to seed the discussion: 1) How can the understanding in the audience be improved? Should one show code at all? What else to show what is going on? 2) Is excluding common synthesis techniques (oscillators, filters) a fruitful limitation or not? 3) How to avoid the problem that Lisp style programs are very easy to write/rewrite, but can be somewhat harder to decipher after a longer time has passed? 4) Algorithmic rewriting of programs

Technical requirements:

- Beamer
- Stereo sound (I can bring my RME interface with 2xJack/XLR output).

Example music:

A piece of music made with Cisp: https://soundcloud.com/sper/translation

Video etudes: https://www.casperschipper.nl/v2/uncategorized/a-few-noisy-etudes-in-cisp/